BOOK 2: MAKING DECISIONS
Chapter 1: The If Statement

WHEN CODE SHOULD RUN SOMETIMES

Until now, every line of code you wrote executed. Line 1 ran, then line 2, then line 3, all the way to the end.

Real programs do not work that way. Real programs make decisions.

If the user is logged in, show their profile.
If the password is wrong, show an error.
If the cart is empty, hide the checkout button.
If it is the user's birthday, show a special message.

The key word is IF. Something happens only IF a condition is true.

JavaScript has a statement for exactly this: the if statement.

BASIC IF STRUCTURE

Here is the simplest if statement:

```
let age = 25

if (age >= 18) {
 console.log("You are an adult")
}
```

The message appears because 25 is greater than or equal to 18. The condition is true.

Change the age:

```
let age = 15

if (age >= 18) {
 console.log("You are an adult")
}
```

Nothing appears. The condition is false, so the code inside is skipped.

The structure has four parts:

The if keyword
A condition in parentheses
An opening curly brace
Code to run if the condition is true
A closing curly brace

If the condition is true, run the code inside the braces. If the condition is false, skip it entirely.

THE CONDITION

The condition must produce true or false. Usually this is a comparison:

```
if (score > 100) {
 console.log("High score!")
}

if (password === "secret123") {
 console.log("Access granted")
}

if (items.length === 0) {
 console.log("Your cart is empty")
```

```
}

if (temperature < 32) {
 console.log("Freezing!")
}
```

Each condition evaluates to true or false. Only true triggers the code inside.

The condition can be any expression that produces a boolean:

```
let isLoggedIn = true

if (isLoggedIn) {
 console.log("Welcome back!")
}
```

The variable itself is the condition. It is already true or false.

You can use logical operators:

```
let age = 25
let hasLicense = true

if (age >= 16 && hasLicense) {
 console.log("You can drive")
}
```

Both must be true. Age must be 16 or more AND hasLicense must be true.

```
let isWeekend = true
let isHoliday = false

if (isWeekend || isHoliday) {
 console.log("No work today!")
}
```

Either can be true. It is the weekend OR it is a holiday.

THE BLOCK

The curly braces define a block of code. Everything inside runs as a group.

```
let hour = 14

if (hour < 12) {
 console.log("Good morning!")
 console.log("Have a great start to your day.")
 console.log("Time for breakfast?")
}
```

If the condition is true, all three messages appear. If false, none of them appear. They are treated as one unit.

You can put any code inside the block. Variables, math, more console.log statements, even other if statements.

```
let score = 85

if (score >= 70) {
 let passed = true
 let grade = "C or better"
 console.log(`You passed!`)
 console.log(`Grade: ${grade}`)
}
```

Multiple statements work together inside the block.

INDENTATION

Notice how the code inside the braces is indented (pushed to the right). This is not required by JavaScript but it is essential for

humans.

Bad:

```
if (age >= 18) {
console.log("Adult")
let canVote = true
console.log("Can vote")
}
```

Hard to see what is inside the if statement.

Good:

```
if (age >= 18) {
 console.log("Adult")
 let canVote = true
 console.log("Can vote")
}
```

Instantly clear what code is controlled by the if statement.

Most programmers use 2 or 4 spaces for indentation. Pick one and be consistent.

EXAMPLES

Check if a number is positive:

```
let number = 42

if (number > 0) {
 console.log(`${number} is positive`)
}
```

Check if a string is empty:

```
let name = ""

if (name.length === 0) {
 console.log("Please enter your name")
}
```

Check if a user can access content:

```
let userAge = 21
let contentRating = 18

if (userAge >= contentRating) {
 console.log("Access granted to this content")
}
```

Check multiple conditions:

```
let hour = 14
let isWeekday = true

if (hour >= 9 && hour < 17 && isWeekday) {
 console.log("Business hours - we are open!")
}
```

COMMON MISTAKES

Forgetting parentheses around the condition:

```
if age >= 18 {
 console.log("Adult")
}
```

This causes an error. The condition must be in parentheses.

Using = instead of === in the condition:

```
let status = "active"
```

```
if (status = "inactive") {
 console.log("Account is inactive")
}
```

This is wrong. Single = assigns a value. It sets status to "inactive" and then the if always runs. Use === to compare.

Putting a semicolon after the condition:

```
if (age >= 18); {
 console.log("Adult")
}
```

The semicolon ends the if statement early. The block always runs regardless of the condition. Do not put a semicolon there.

PRACTICE

Write an if statement that checks if a temperature is above 100 and displays "Fever detected!"

Write an if statement that checks if a string has more than 10 characters and displays "Long text"

Write an if statement that checks if a number is even (hint: use % 2 === 0) and displays "Even number"

Write an if statement with two conditions using && that checks if someone is old enough (18+) and has enough money (at least $100) to display "You can make this purchase"

WHAT YOU LEARNED

You learned that if statements run code only when a condition is true.
You learned the structure: if (condition) { code }
You learned that conditions must produce true or false.
You learned that blocks group multiple statements together.
You learned to indent code inside blocks for readability.
You learned common mistakes to avoid.

The next chapter teaches you what to do when the condition is false.

BOOK 2: MAKING DECISIONS
Chapter 2: If and Else

TWO PATHS

Sometimes you want to do one thing if a condition is true, and a different thing if it is false. This is where else comes in.

```
let age = 15

if (age >= 18) {
 console.log("You are an adult")
} else {
 console.log("You are a minor")
}
```

Since age is 15, which is not greater than or equal to 18, the else block runs. Output: "You are a minor"

Change the age:

```
let age = 25
```

```
if (age >= 18) {
 console.log("You are an adult")
} else {
 console.log("You are a minor")
}
```

Now age is 25, which is greater than or equal to 18. The if block runs. Output: "You are an adult"

The else block runs when the if condition is false. One or the other always runs, never both, never neither.

WHEN IF IS TRUE

When the condition is true, JavaScript:

1. Runs all the code in the if block
2. Skips the else block completely
3. Continues after the closing brace

```
let score = 85

if (score >= 70) {
 console.log("You passed!")
 console.log("Congratulations!")
} else {
 console.log("You failed")
 console.log("Please try again")
}

console.log("Thank you for taking the test")
```

Output:
You passed!
Congratulations!
Thank you for taking the test

The else block was completely skipped. The final message runs regardless because it is outside both blocks.

WHEN IF IS FALSE

When the condition is false, JavaScript:

1. Skips the if block completely
2. Runs all the code in the else block
3. Continues after the closing brace

```
let score = 55

if (score >= 70) {
 console.log("You passed!")
 console.log("Congratulations!")
} else {
 console.log("You failed")
 console.log("Please try again")
}

console.log("Thank you for taking the test")
```

Output:
You failed
Please try again
Thank you for taking the test

The if block was completely skipped. The final message still runs.

ONLY ONE PATH RUNS

This is important: exactly one path runs. Never both. Never neither.

```
let coin = "heads"

if (coin === "heads") {
 console.log("You win!")
} else {
 console.log("You lose!")
}
```

If coin is "heads", you see "You win!" and nothing else.
If coin is anything else, you see "You lose!" and nothing else.

This is like a fork in the road. You must go left or right. You cannot go both ways. You cannot stand still.

PRACTICAL EXAMPLES

Login check:

```
let password = "secret123"
let correctPassword = "secret123"

if (password === correctPassword) {
 console.log("Welcome! You are now logged in.")
} else {
 console.log("Incorrect password. Please try again.")
}
```

Cart check:

```
let cartItems = 0

if (cartItems > 0) {
 console.log(`You have ${cartItems} items in your cart`)
 console.log("Proceed to checkout?")
} else {
 console.log("Your cart is empty")
 console.log("Start shopping!")
```

```
}
```

Time of day greeting:

```
let hour = 14

if (hour < 12) {
 console.log("Good morning!")
} else {
 console.log("Good afternoon!")
}
```

Even or odd:

```
let number = 7

if (number % 2 === 0) {
 console.log(`${number} is even`)
} else {
 console.log(`${number} is odd`)
}
```

Discount eligibility:

```
let purchaseAmount = 75
let discountThreshold = 100

if (purchaseAmount >= discountThreshold) {
 let discount = purchaseAmount * 0.1
 console.log(`You qualify for a discount of $${discount}`)
} else {
 let remaining = discountThreshold - purchaseAmount
 console.log(`Spend $${remaining} more to get 10% off`)
}
```

STORING RESULTS

You can use if/else to set variables:

```
let temperature = 45
let weatherDescription

if (temperature > 70) {
 weatherDescription = "warm"
} else {
 weatherDescription = "cold"
}

console.log(`It is ${weatherDescription} today`)
```

Notice we declared weatherDescription before the if/else. Variables created inside a block only exist inside that block. If we wrote:

```
if (temperature > 70) {
 let weatherDescription = "warm"
} else {
 let weatherDescription = "cold"
}

console.log(weatherDescription)
```

This would cause an error. The variable does not exist outside the blocks.

Declare the variable first, then assign it inside the blocks.

BLOCKS WITH ONE STATEMENT

If your block has only one statement, you can technically skip the braces:

```
if (age >= 18)
 console.log("Adult")
else
```

```
 console.log("Minor")
```

This works but is dangerous. If you add a second line later:

```
if (age >= 18)
 console.log("Adult")
 console.log("You can vote")
else
 console.log("Minor")
```

This causes an error. Only the first console.log is part of the if. The second runs unconditionally, and the else has nothing to attach to.

Always use braces, even for single statements:

```
if (age >= 18) {
 console.log("Adult")
} else {
 console.log("Minor")
}
```

Clear, safe, and easy to add more code later.

PRACTICE

Write an if/else that checks if a number is positive or negative and displays the appropriate message.

Write an if/else that checks if a password has at least 8 characters. If yes, display "Password is strong enough". If no, display "Password is too short".

Write an if/else that checks if a user's balance is at least $50. If yes, calculate and display a 5% reward. If no, display how much more they need.

Write an if/else that checks if the current hour is between 9 and 17

(inclusive). Display "Open" or "Closed" appropriately.

WHAT YOU LEARNED

You learned that else provides an alternative when if is false.
You learned that exactly one block runs, never both.
You learned to declare variables before if/else when you need them after.
You learned to always use braces even for single statements.
You learned common patterns like login checks and threshold comparisons.

The next chapter teaches you how to handle more than two options.

BOOK 2: MAKING DECISIONS

Chapter 3: Multiple Paths

MORE THAN TWO OPTIONS

Sometimes you need more than two choices. Is it morning, afternoon, or evening? Is the grade A, B, C, D, or F? Is the traffic light red, yellow, or green?

The else if statement handles this:

```
let hour = 14

if (hour < 12) {
 console.log("Good morning!")
} else if (hour < 17) {
 console.log("Good afternoon!")
} else {
 console.log("Good evening!")
}
```

Output: Good afternoon!

The hour is 14. First check: is 14 less than 12? No. Move to else if: is 14 less than 17? Yes. Run that block. Skip everything else.

Change the hour to see different paths:

hour = 9 gives "Good morning!" (first condition is true)
hour = 14 gives "Good afternoon!" (second condition is true)
hour = 20 gives "Good evening!" (both conditions false, else runs)

HOW ELSE IF WORKS

JavaScript checks conditions from top to bottom. It stops at the first true condition.

```
let score = 85

if (score >= 90) {
 console.log("Grade: A")
} else if (score >= 80) {
 console.log("Grade: B")
} else if (score >= 70) {
 console.log("Grade: C")
} else if (score >= 60) {
 console.log("Grade: D")
} else {
 console.log("Grade: F")
}
```

Output: Grade: B

Score is 85. First check: is 85 >= 90? No. Second check: is 85 >= 80? Yes. Print "Grade: B" and stop. The remaining conditions are never checked.

This is important: only ONE block runs. Even if multiple conditions would be true, JavaScript stops at the first match.

```
let score = 95

if (score >= 90) {
 console.log("Grade: A")
} else if (score >= 80) {
 console.log("Grade: B")
} else if (score >= 70) {
 console.log("Grade: C")
}
```

Score 95 is greater than 90, 80, and 70. But only "Grade: A" prints because JavaScript stops after the first true condition.

ORDER MATTERS

Because JavaScript stops at the first true condition, order is crucial.

Wrong order:

```
let score = 95

if (score >= 60) {
 console.log("Grade: D")
} else if (score >= 70) {
 console.log("Grade: C")
} else if (score >= 80) {
 console.log("Grade: B")
} else if (score >= 90) {
 console.log("Grade: A")
}
```

Output: Grade: D

This is wrong! 95 is greater than 60, so the first condition is true.

JavaScript stops there and never checks the higher grades.

Always check the most specific (or highest) condition first:

```
let score = 95

if (score >= 90) {
 console.log("Grade: A")
} else if (score >= 80) {
 console.log("Grade: B")
} else if (score >= 70) {
 console.log("Grade: C")
} else if (score >= 60) {
 console.log("Grade: D")
} else {
 console.log("Grade: F")
}
```

Now it correctly outputs "Grade: A".

THE FINAL ELSE

The else at the end catches anything that did not match the previous conditions.

```
let weather = "snowing"

if (weather === "sunny") {
 console.log("Wear sunglasses")
} else if (weather === "raining") {
 console.log("Bring an umbrella")
} else if (weather === "snowing") {
 console.log("Wear a coat")
} else {
 console.log("Check the weather app")
}
```

Output: Wear a coat

The else is a safety net. If you add a new weather type later and forget to add a condition, the else handles it gracefully.

You do not always need a final else:

```
let userRole = "admin"

if (userRole === "admin") {
 console.log("Full access granted")
} else if (userRole === "editor") {
 console.log("Edit access granted")
} else if (userRole === "viewer") {
 console.log("View access granted")
}
```

If userRole is something unexpected like "guest", nothing happens. Sometimes that is fine. Other times you want explicit handling. Consider your needs.

MULTIPLE CONDITIONS IN EACH CHECK

You can combine conditions with && and || inside each if or else if:

```
let age = 25
let hasTicket = true

if (age < 12 && hasTicket) {
 console.log("Child admission: $5")
} else if (age < 18 && hasTicket) {
 console.log("Youth admission: $10")
} else if (age >= 18 && hasTicket) {
 console.log("Adult admission: $15")
} else {
 console.log("Please purchase a ticket")
}
```

Output: Adult admission: $15

Each condition can be as simple or complex as needed.

```
let day = "Saturday"
let hour = 10

if (day === "Saturday" || day === "Sunday") {
 if (hour < 12) {
 console.log("Weekend morning")
 } else {
 console.log("Weekend afternoon/evening")
 }
} else {
 console.log("Weekday")
}
```

Output: Weekend morning

PRACTICAL EXAMPLES

Traffic light:

```
let light = "yellow"

if (light === "green") {
 console.log("Go")
} else if (light === "yellow") {
 console.log("Slow down")
} else if (light === "red") {
 console.log("Stop")
} else {
 console.log("Invalid light color")
}
```

Temperature description:

```
let temperature = 72

if (temperature >= 90) {
 console.log("Hot")
} else if (temperature >= 70) {
 console.log("Warm")
} else if (temperature >= 50) {
 console.log("Cool")
} else if (temperature >= 32) {
 console.log("Cold")
} else {
 console.log("Freezing")
}
```

User subscription tier:

```
let monthlySpend = 75

if (monthlySpend >= 100) {
 console.log("Platinum member - 20% discount")
} else if (monthlySpend >= 50) {
 console.log("Gold member - 10% discount")
} else if (monthlySpend >= 25) {
 console.log("Silver member - 5% discount")
} else {
 console.log("Bronze member - no discount")
}
```

Password strength checker:

```
let password = "MyP@ssw0rd"
let length = password.length

if (length >= 16) {
 console.log("Very strong password")
} else if (length >= 12) {
 console.log("Strong password")
} else if (length >= 8) {
 console.log("Acceptable password")
```

```
} else {
 console.log("Password too short")
}
```

HTTP status codes:

```
let statusCode = 404

if (statusCode >= 200 && statusCode < 300) {
 console.log("Success")
} else if (statusCode >= 300 && statusCode < 400) {
 console.log("Redirect")
} else if (statusCode >= 400 && statusCode < 500) {
 console.log("Client error")
} else if (statusCode >= 500) {
 console.log("Server error")
} else {
 console.log("Unknown status")
}
```

STORING RESULTS FROM MULTIPLE PATHS

Just like with if/else, declare the variable before the chain:

```
let score = 85
let letterGrade

if (score >= 90) {
 letterGrade = "A"
} else if (score >= 80) {
 letterGrade = "B"
} else if (score >= 70) {
 letterGrade = "C"
} else if (score >= 60) {
 letterGrade = "D"
} else {
 letterGrade = "F"
```

```
}

console.log(`Your grade is ${letterGrade}`)
```

The variable letterGrade exists after the chain and holds the assigned value.

PRACTICE

Write an else if chain that takes a number 1-12 and displays the month name. Handle invalid numbers.

Write an else if chain for a restaurant tip calculator: 0-10% is poor, 11-15% is standard, 16-20% is good, over 20% is excellent.

Write an else if chain for shipping costs: orders under $25 pay $8, orders $25-49 pay $5, orders $50-99 pay $3, orders $100+ ship free.

Write an else if chain for a simple game: rock beats scissors, scissors beats paper, paper beats rock. Handle ties.

WHAT YOU LEARNED

You learned that else if adds more options beyond two paths.
You learned that JavaScript checks conditions top to bottom and stops at the first true one.
You learned that order matters because later conditions are skipped after a match.
You learned that a final else catches anything that did not match.
You learned to use else if chains for categorizing values.

The next chapter teaches you how to nest conditions inside other conditions.

BOOK 2: MAKING DECISIONS

Chapter 4: Nested Conditions

CONDITIONS INSIDE CONDITIONS

Sometimes you need to make a decision, and then make another decision based on the first one. This is nesting: putting one if statement inside another.

```
let isLoggedIn = true
let isAdmin = true

if (isLoggedIn) {
 console.log("Welcome back!")

 if (isAdmin) {
 console.log("You have admin access")
 }
}
```

Output:
Welcome back!
You have admin access

The outer if checks if the user is logged in. Only if that is true does JavaScript even look at the inner if checking for admin status.

Change isLoggedIn to false:

```
let isLoggedIn = false
let isAdmin = true

if (isLoggedIn) {
 console.log("Welcome back!")

 if (isAdmin) {
```

```
 console.log("You have admin access")
 }
}
```

No output. The outer condition is false, so everything inside is skipped. The inner if is never even evaluated.

WHY NEST CONDITIONS

Nesting makes sense when the second decision only matters if the first decision goes a certain way.

```
let hasAccount = true
let emailVerified = false

if (hasAccount) {
 if (emailVerified) {
 console.log("Full access granted")
 } else {
 console.log("Please verify your email")
 }
} else {
 console.log("Please create an account")
}
```

Output: Please verify your email

This creates a logical flow:
1. First check: do they have an account?
2. If no: tell them to create one
3. If yes: check if their email is verified
4. Based on verification: grant access or ask them to verify

Without nesting, you would need more complex conditions.

NESTED ELSE IF

You can nest entire if/else if/else chains:

```
let userType = "premium"
let contentRating = "R"

if (userType === "premium") {
 console.log("Premium user detected")

 if (contentRating === "G") {
 console.log("Access granted to all content")
 } else if (contentRating === "PG") {
 console.log("Access granted to all content")
 } else if (contentRating === "R") {
 console.log("Access granted - mature content warning shown")
 }
} else if (userType === "basic") {
 console.log("Basic user detected")

 if (contentRating === "G") {
 console.log("Access granted")
 } else {
 console.log("Upgrade to premium for this content")
 }
} else {
 console.log("Please log in")
}
```

Output:
Premium user detected
Access granted - mature content warning shown

DEEP NESTING

You can nest multiple levels deep:

```
let country = "USA"
let state = "California"
let city = "Los Angeles"

if (country === "USA") {
 console.log("United States")

 if (state === "California") {
 console.log("California")

 if (city === "Los Angeles") {
 console.log("Los Angeles - Welcome to LA!")
 } else if (city === "San Francisco") {
 console.log("San Francisco - Enjoy the bay!")
 } else {
 console.log(`Welcome to ${city}`)
 }
 } else if (state === "Texas") {
 console.log("Texas - Everything is bigger here")
 }
} else if (country === "Canada") {
 console.log("Canada - Welcome, eh!")
}
```

Output:
United States
California
Los Angeles - Welcome to LA!

But be careful. Deep nesting becomes hard to read. More than 2-3 levels is usually a sign you should restructure your code.

AVOIDING DEEP NESTING

Instead of nesting deeply, you can sometimes combine conditions:

Deeply nested (hard to read):

```
if (isLoggedIn) {
 if (hasPermission) {
 if (resourceExists) {
 console.log("Showing resource")
 }
 }
}
```

Combined conditions (easier to read):

```
if (isLoggedIn && hasPermission && resourceExists) {
 console.log("Showing resource")
}
```

Both do the same thing. The second is cleaner.

Another technique is early returns (covered when we learn functions). For now, know that deep nesting is often avoidable.

NESTING WITH ELSE

Each if in a nest can have its own else:

```
let age = 25
let hasID = true

if (age >= 21) {
 if (hasID) {
 console.log("Entry permitted")
 } else {
 console.log("Please show your ID")
 }
} else {
 console.log("Must be 21 or older")
}
```

Output: Entry permitted

The outer if has an else for underage users. The inner if has an else for people without ID.

PRACTICAL EXAMPLES

Shopping cart checkout:

```
let itemsInCart = 3
let isLoggedIn = true
let hasPaymentMethod = true

if (itemsInCart > 0) {
 console.log(`${itemsInCart} items in cart`)

 if (isLoggedIn) {
 if (hasPaymentMethod) {
 console.log("Ready to checkout!")
 } else {
 console.log("Please add a payment method")
 }
 } else {
 console.log("Please log in to checkout")
 }
} else {
 console.log("Your cart is empty")
}
```

Game level unlock:

```
let currentLevel = 5
let hasKey = true
let bossDefeated = true

if (currentLevel >= 5) {
```

```
 console.log("Approaching secret area...")

 if (hasKey) {
 if (bossDefeated) {
 console.log("Secret area unlocked!")
 } else {
 console.log("Defeat the boss first")
 }
 } else {
 console.log("You need the golden key")
 }
} else {
 console.log("Reach level 5 to access this area")
}
```

Form validation:

```
let username = "alex123"
let password = "secretpass"
let confirmPassword = "secretpass"

if (username.length >= 3) {
 console.log("Username valid")

 if (password.length >= 8) {
 console.log("Password length valid")

 if (password === confirmPassword) {
 console.log("Passwords match - account created!")
 } else {
 console.log("Passwords do not match")
 }
 } else {
 console.log("Password must be at least 8 characters")
 }
} else {
 console.log("Username must be at least 3 characters")
}
```

Time and day based greeting:

```
let day = "Saturday"
let hour = 10

if (day === "Saturday" || day === "Sunday") {
 console.log("It is the weekend!")

 if (hour < 12) {
 console.log("Enjoy your morning sleep-in")
 } else {
 console.log("Enjoy your afternoon")
 }
} else {
 console.log("It is a weekday")

 if (hour >= 9 && hour < 17) {
 console.log("Working hours")
 } else if (hour < 9) {
 console.log("Getting ready for work")
 } else {
 console.log("After work hours")
 }
}
```

INDENTATION IS CRITICAL

Proper indentation shows the structure of nested code:

Bad indentation (confusing):

```
if (a) {
if (b) {
console.log("Both true")
} else {
console.log("a true, b false")
}
} else {
```

```
console.log("a false")
}
```

Good indentation (clear):

```
if (a) {
 if (b) {
 console.log("Both true")
 } else {
 console.log("a true, b false")
 }
} else {
 console.log("a false")
}
```

Each level of nesting adds one level of indentation. This makes the logic visible at a glance.

PRACTICE

Write nested conditions for a movie theater: check if the user is old enough for the movie rating (G, PG, PG-13, R), and if old enough, check if they have a ticket.

Write nested conditions for an online store: check if the item is in stock, then check if the user is logged in, then check if they have enough balance.

Write nested conditions for a game: check if the player has enough health to continue, then check if they have a weapon equipped, then check if they have ammunition.

WHAT YOU LEARNED

You learned that you can put if statements inside other if statements.
You learned that inner conditions only run if outer conditions are true.
You learned that nesting is useful when decisions depend on previous decisions.
You learned that deep nesting should be avoided for readability.
You learned that proper indentation is essential for nested code.

The next chapter teaches you an alternative to long else if chains: the switch statement.

BOOK 2: MAKING DECISIONS

Chapter 5: The Switch Statement

WHEN YOU HAVE MANY EXACT MATCHES

Sometimes you check one value against many possible exact matches:

```
let day = "Wednesday"

if (day === "Monday") {
 console.log("Start of the work week")
} else if (day === "Tuesday") {
 console.log("Second day")
} else if (day === "Wednesday") {
 console.log("Midweek")
} else if (day === "Thursday") {
 console.log("Almost Friday")
} else if (day === "Friday") {
 console.log("End of work week")
} else if (day === "Saturday") {
 console.log("Weekend!")
} else if (day === "Sunday") {
 console.log("Weekend!")
} else {
```

```
 console.log("Invalid day")
}
```

This works but is repetitive. The switch statement handles this pattern more cleanly:

```
let day = "Wednesday"

switch (day) {
 case "Monday":
 console.log("Start of the work week")
 break
 case "Tuesday":
 console.log("Second day")
 break
 case "Wednesday":
 console.log("Midweek")
 break
 case "Thursday":
 console.log("Almost Friday")
 break
 case "Friday":
 console.log("End of work week")
 break
 case "Saturday":
 console.log("Weekend!")
 break
 case "Sunday":
 console.log("Weekend!")
 break
 default:
 console.log("Invalid day")
}
```

Output: Midweek

SWITCH STRUCTURE

The switch statement has these parts:

```
switch (value) {
 case match1:
 code to run if value equals match1
 break
 case match2:
 code to run if value equals match2
 break
 default:
 code to run if no case matches
}
```

The value in parentheses is compared against each case. When a match is found, that code runs. The break statement stops execution and exits the switch.

The default section runs if no case matches. It is like the final else in an if/else chain. It is optional but usually a good idea to include.

THE BREAK KEYWORD

Break is crucial. Without it, JavaScript continues running into the next case:

```
let fruit = "apple"

switch (fruit) {
 case "apple":
 console.log("It is an apple")
 case "banana":
 console.log("It is a banana")
 case "orange":
 console.log("It is an orange")
}
```

Output:
It is an apple
It is a banana
It is an orange

This is called "fall through". JavaScript matched "apple" and then ran every case after it because there were no breaks.

Always use break unless you intentionally want fall through:

```
let fruit = "apple"

switch (fruit) {
 case "apple":
 console.log("It is an apple")
 break
 case "banana":
 console.log("It is a banana")
 break
 case "orange":
 console.log("It is an orange")
 break
}
```

Output: It is an apple

INTENTIONAL FALL THROUGH

Sometimes fall through is useful. When multiple cases should do the same thing:

```
let day = "Saturday"

switch (day) {
 case "Saturday":
 case "Sunday":
 console.log("Weekend!")
```

```
 break
 case "Monday":
 case "Tuesday":
 case "Wednesday":
 case "Thursday":
 case "Friday":
 console.log("Weekday")
 break
 default:
 console.log("Invalid day")
}
```

Output: Weekend!

Saturday and Sunday share the same code. Monday through Friday share different code. The empty cases fall through to the next break.

SWITCH WITH NUMBERS

Switch works with numbers too:

```
let month = 3

switch (month) {
 case 1:
 console.log("January")
 break
 case 2:
 console.log("February")
 break
 case 3:
 console.log("March")
 break
 case 4:
 console.log("April")
 break
 case 5:
```

```
 console.log("May")
 break
 case 6:
 console.log("June")
 break
 case 7:
 console.log("July")
 break
 case 8:
 console.log("August")
 break
 case 9:
 console.log("September")
 break
 case 10:
 console.log("October")
 break
 case 11:
 console.log("November")
 break
 case 12:
 console.log("December")
 break
 default:
 console.log("Invalid month")
}
```

Output: March

SWITCH USES STRICT EQUALITY

Switch compares using strict equality (===). Types must match:

```
let value = "5"

switch (value) {
 case 5:
```

```
 console.log("Number five")
 break
 case "5":
 console.log("String five")
 break
}
```

Output: String five

The string "5" matches the case "5", not the case 5.

PRACTICAL EXAMPLES

Command processor:

```
let command = "start"

switch (command) {
 case "start":
 console.log("Starting the application...")
 break
 case "stop":
 console.log("Stopping the application...")
 break
 case "restart":
 console.log("Restarting the application...")
 break
 case "status":
 console.log("Checking status...")
 break
 default:
 console.log("Unknown command. Try: start, stop, restart, status")
}
```

Grade to description:

```
let grade = "B"
```

```
switch (grade) {
 case "A":
 console.log("Excellent!")
 break
 case "B":
 console.log("Good job!")
 break
 case "C":
 console.log("Satisfactory")
 break
 case "D":
 console.log("Needs improvement")
 break
 case "F":
 console.log("Failed")
 break
 default:
 console.log("Invalid grade")
}
```

HTTP method handler:

```
let method = "POST"

switch (method) {
 case "GET":
 console.log("Fetching data...")
 break
 case "POST":
 console.log("Creating new data...")
 break
 case "PUT":
 console.log("Updating data...")
 break
 case "DELETE":
 console.log("Deleting data...")
 break
 default:
 console.log("Method not allowed")
```

```
}
```

Days in a month (using fall through):

```
let month = 2
let year = 2024
let days

switch (month) {
 case 1:
 case 3:
 case 5:
 case 7:
 case 8:
 case 10:
 case 12:
 days = 31
 break
 case 4:
 case 6:
 case 9:
 case 11:
 days = 30
 break
 case 2:
 days = 28
 break
 default:
 days = 0
}

console.log(`Month ${month} has ${days} days`)
```

Calculator operation:

```
let operation = "*"
let a = 10
let b = 5
let result
```

```
switch (operation) {
 case "+":
 result = a + b
 break
 case "-":
 result = a - b
 break
 case "*":
 result = a * b
 break
 case "/":
 result = a / b
 break
 default:
 console.log("Unknown operation")
}

console.log(`${a} ${operation} ${b} = ${result}`)
```

WHEN TO USE SWITCH VS IF/ELSE

Use switch when:
- Comparing one value against many exact matches
- The matches are constants (specific values like "Monday" or 5)
- You have more than 3-4 options

Use if/else when:
- Checking ranges (score >= 90)
- Checking multiple different conditions
- Using complex boolean expressions
- You have only 2-3 options

Switch cannot do:

```
switch (score) {
 case >= 90:
 console.log("A")
```

```
 break
}
```

This is invalid. Switch only does exact equality. For ranges, use if/else if.

MULTIPLE STATEMENTS IN A CASE

You can run multiple statements in each case:

```
let status = "processing"

switch (status) {
 case "pending":
 console.log("Order received")
 console.log("Waiting for payment")
 break
 case "processing":
 console.log("Payment confirmed")
 console.log("Preparing your order")
 console.log("Estimated delivery: 3-5 days")
 break
 case "shipped":
 console.log("Your order is on the way!")
 console.log("Track your package online")
 break
 case "delivered":
 console.log("Order delivered")
 console.log("Thank you for shopping with us!")
 break
 default:
 console.log("Unknown status")
}
```

Output:
Payment confirmed
Preparing your order

Estimated delivery: 3-5 days

PRACTICE

Write a switch for compass directions (N, S, E, W) that displays the full direction name.

Write a switch for a traffic light (red, yellow, green) that displays what the driver should do.

Write a switch for file extensions (txt, pdf, jpg, mp3) that displays the file type (text, document, image, audio).

Write a switch for keyboard shortcuts (ctrl + s, ctrl + c, ctrl + v, ctrl + z) that displays the action (save, copy, paste, undo).

WHAT YOU LEARNED

You learned that switch compares one value against multiple cases.
You learned that break exits the switch after a case runs.
You learned that without break, execution falls through to the next case.
You learned that default handles unmatched cases.
You learned when to use switch versus if/else.

The next chapter teaches you a shorthand for simple if/else: the ternary operator.
BOOK 2: MAKING DECISIONS
Chapter 6: The Ternary Operator

A SHORTHAND FOR IF/ELSE

Consider this common pattern:

```
let age = 20
let status

if (age >= 18) {
 status = "adult"
} else {
 status = "minor"
}

console.log(status)
```

Five lines just to assign one of two values. The ternary operator does this in one line:

```
let age = 20
let status = age >= 18 ? "adult" : "minor"

console.log(status)
```

Output: adult

Same result, much shorter.

THE STRUCTURE

```
condition ? valueIfTrue : valueIfFalse
```

The ternary operator has three parts:

1. A condition (before the question mark)
2. A value if the condition is true (between ? and :)
3. A value if the condition is false (after the colon)

```
let temperature = 75
let weather = temperature > 80 ? "hot" : "comfortable"

console.log(weather)
```

Output: comfortable

Temperature is 75. Is 75 greater than 80? No. So weather gets the value after the colon: "comfortable".

Change temperature to 85:

```
let temperature = 85
let weather = temperature > 80 ? "hot" : "comfortable"
```

Output: hot

Now 85 is greater than 80. Weather gets the value after the question mark: "hot".

USING TERNARY DIRECTLY

You do not need to store the result in a variable. Use it anywhere a value is needed:

```
let age = 20
console.log(age >= 18 ? "adult" : "minor")
```

Output: adult

Inside template literals:

```
let score = 85
console.log(`You ${score >= 70 ? "passed" : "failed"} the test`)
```

Output: You passed the test

In math:

```
let isPremium = true
let price = 100
let finalPrice = isPremium ? price * 0.9 : price

console.log(finalPrice)
```

Output: 90

Premium members get 10% off.

TERNARY VS IF/ELSE

Ternary is best for simple value assignments. These are equivalent:

If/else version:

```
let message
if (isLoggedIn) {
 message = "Welcome back!"
} else {
 message = "Please log in"
}
```

Ternary version:

```
let message = isLoggedIn ? "Welcome back!" : "Please log in"
```

Use ternary when:
- Choosing between two values
- The logic is simple and clear
- It fits on one line comfortably

Use if/else when:
- You need to run multiple statements
- The logic is complex

- The ternary becomes hard to read

AVOID COMPLEX TERNARIES

Ternary should be simple. This is hard to read:

```
let result = a > b ? (a > c ? a : c) : (b > c ? b : c)
```

This nests ternaries inside ternaries. It works but is confusing. Write it as if/else instead:

```
let result
if (a > b) {
 if (a > c) {
 result = a
 } else {
 result = c
 }
} else {
 if (b > c) {
 result = b
 } else {
 result = c
 }
}
```

Or even better, use Math.max:

```
let result = Math.max(a, b, c)
```

If you struggle to understand a ternary at a glance, it is too complex.

PRACTICAL EXAMPLES

Toggle text:

```
let isOpen = false
let buttonText = isOpen ? "Close" : "Open"

console.log(buttonText)
```

Display quantity:

```
let count = 1
let text = count === 1 ? "item" : "items"

console.log(`${count} ${text} in cart`)
```

Output: 1 item in cart

```
let count = 5
let text = count === 1 ? "item" : "items"

console.log(`${count} ${text} in cart`)
```

Output: 5 items in cart

Show or hide element (conceptually):

```
let hasNotifications = true
let notificationBadge = hasNotifications ? "visible" : "hidden"
```

Greeting based on time:

```
let hour = 14
let greeting = hour < 12 ? "Good morning" : "Good afternoon"

console.log(greeting)
```

Discount calculation:

```
let isMember = true
let purchaseAmount = 100
```

```
let discount = isMember ? 0.15 : 0

let finalAmount = purchaseAmount * (1 - discount)

console.log(`Total: $${finalAmount}`)
```

Output: Total: $85

Age verification:

```
let age = 17
let canEnter = age >= 21 ? "allowed" : "denied"

console.log(`Entry ${canEnter}`)
```

Null or undefined check:

```
let username = null
let displayName = username ? username : "Guest"

console.log(`Hello, ${displayName}`)
```

Output: Hello, Guest

TERNARY WITH FUNCTION CALLS

The values can be function calls or any expression:

```
let isDebug = true

isDebug ? console.log("Debug mode") : console.log("Production
mode")
```

Output: Debug mode

This works but is unusual. Most people would write:

```
if (isDebug) {
 console.log("Debug mode")
} else {
 console.log("Production mode")
}
```

Use ternary for values, if/else for actions.

COMMON PATTERNS

Default value:

```
let input = ""
let value = input ? input : "default"
```

This checks if input is truthy. If not, use "default".

Conditional class name (in web development):

```
let isActive = true
let className = isActive ? "button-active" : "button-inactive"
```

Conditional styling:

```
let isDarkMode = true
let backgroundColor = isDarkMode ? "#333" : "#fff"
let textColor = isDarkMode ? "#fff" : "#333"
```

Pluralization:

```
let commentCount = 1
let label = commentCount === 1 ? "comment" : "comments"

console.log(`${commentCount} ${label}`)
```

Sign indicator:

```
let balance = -50
let indicator = balance >= 0 ? "+" : ""

console.log(`Balance: ${indicator}${balance}`)
```

Output: Balance: -50

(Positive numbers would show +50)

PRACTICE

Write a ternary that sets a variable to "even" if a number is even, "odd" if not.

Write a ternary that sets a greeting to "Good morning" if the hour is less than 12, otherwise "Good evening".

Write a ternary that calculates a tip: 20% if service was "excellent", 15% otherwise.

Write a ternary that sets a message to "Empty" if an array has 0 items, "Has items" otherwise.

WHAT YOU LEARNED

You learned that the ternary operator is shorthand for simple if/else.
You learned the structure: condition ? valueIfTrue : valueIfFalse
You learned to use ternary for value assignment, if/else for complex logic.
You learned to avoid nested or complex ternaries.
You learned common patterns like default values and pluralization.

The next chapter teaches you about truthy and falsy values.

BOOK 2: MAKING DECISIONS

Chapter 7: Truthy and Falsy

NOT EVERYTHING IS TRUE OR FALSE

In JavaScript, conditions do not have to be actual booleans. Any value can be used as a condition, and JavaScript will treat it as either "truthy" or "falsy".

```
let name = "Alex"

if (name) {
 console.log("Name exists")
}
```

Output: Name exists

The variable name is a string, not a boolean. But JavaScript treats non-empty strings as truthy, so the condition passes.

```
let name = ""

if (name) {
 console.log("Name exists")
} else {
 console.log("Name is empty")
}
```

Output: Name is empty

An empty string is falsy. The condition fails.

THE FALSY VALUES

JavaScript has exactly six falsy values:

false - the boolean false
0 - the number zero
"" - an empty string (no characters)
null - represents "no value"
undefined - variable declared but not assigned
NaN - "Not a Number" (result of invalid math)

Every other value is truthy.

Check each one:

```
if (false) {
 console.log("This never runs")
}

if (0) {
 console.log("This never runs")
}

if ("") {
 console.log("This never runs")
}

if (null) {
 console.log("This never runs")
}

if (undefined) {
 console.log("This never runs")
}

if (NaN) {
 console.log("This never runs")
}
```

None of these print anything. All conditions are falsy.

EVERYTHING ELSE IS TRUTHY

Any value not in the falsy list is truthy:

```
if (true) {
 console.log("Truthy")
}

if (1) {
 console.log("Truthy")
}

if (-1) {
 console.log("Truthy")
}

if (0.5) {
 console.log("Truthy")
}

if ("hello") {
 console.log("Truthy")
}

if ("0") {
 console.log("Truthy")
}

if (" ") {
 console.log("Truthy")
}

if ([]) {
 console.log("Truthy")
}

if ({}) {
```

```
 console.log("Truthy")
}
```

All of these print "Truthy".

Notice some surprises:

The string "0" is truthy (it is not empty)
The string " " (a space) is truthy (it has a character)
Empty arrays [] are truthy
Empty objects {} are truthy
Negative numbers are truthy
Any non-zero number is truthy

WHY THIS MATTERS

Truthy and falsy let you write shorter conditions:

Instead of:

```
if (username !== "" && username !== null && username !== undefined) {
 console.log(`Hello, ${username}`)
}
```

You can write:

```
if (username) {
 console.log(`Hello, ${username}`)
}
```

If username is empty, null, or undefined, the condition is falsy. If username has any real value, the condition is truthy.

Checking if an array has items:

```
let items = ["apple", "banana"]
```

```
if (items.length) {
 console.log(`You have ${items.length} items`)
} else {
 console.log("Your cart is empty")
}
```

If the array is empty, items.length is 0, which is falsy. If the array has items, the length is truthy.

Checking for user input:

```
let userInput = ""

if (userInput) {
 console.log(`You entered: ${userInput}`)
} else {
 console.log("Please enter something")
}
```

Output: Please enter something

CONVERTING TO BOOLEAN

You can explicitly convert any value to a boolean:

Using Boolean():

```
console.log(Boolean("hello"))
console.log(Boolean(""))
console.log(Boolean(42))
console.log(Boolean(0))
console.log(Boolean(null))
console.log(Boolean([]))
```

Output:
true

false
true
false
false
true

Using double NOT (!!):

```
console.log(!!"hello")
console.log(!!"")
console.log(!!42)
console.log(!!0)
```

Output:
true
false
true
false

The first ! converts to boolean and flips it. The second ! flips it back. The result is the truthy/falsy value as an actual boolean.

THE NOT OPERATOR AND TRUTHY/FALSY

The NOT operator (!) converts the value and flips it:

```
console.log(!true)
console.log(!false)
console.log(!"hello")
console.log(!"")
console.log(!0)
console.log(!1)
console.log(!null)
```

Output:
false
true

false
true
true
false
true

If the value is truthy, !value is false.
If the value is falsy, !value is true.

DEFAULT VALUES WITH OR

A common pattern uses the OR operator (||) to provide defaults:

```
let username = ""
let displayName = username || "Guest"

console.log(displayName)
```

Output: Guest

How this works: if username is truthy, displayName gets that value. If username is falsy (like an empty string), displayName gets "Guest".

```
let username = "Alex"
let displayName = username || "Guest"

console.log(displayName)
```

Output: Alex

Now username is truthy, so it is used.

This is called short-circuit evaluation. OR stops at the first truthy value:

```
let a = "" || "default"
```

```
let b = null || "default"
let c = undefined || "default"
let d = "value" || "default"

console.log(a)
console.log(b)
console.log(c)
console.log(d)
```

Output:
default
default
default
value

CHECKING WITH AND

The AND operator (&&) can also use truthy/falsy:

```
let user = "Alex"
user && console.log(`Welcome, ${user}`)
```

Output: Welcome, Alex

If user is truthy, JavaScript continues to the right side. If user is falsy, it stops immediately.

```
let user = ""
user && console.log(`Welcome, ${user}`)
```

No output. User is falsy, so the right side never runs.

This is less common than using if, but you will see it in code.

WATCH OUT FOR ZERO

A common bug involves zero:

```
let quantity = 0
let message = quantity || "No items"

console.log(message)
```

Output: No items

But wait. Zero is a valid quantity. You might want to show "0 items", not "No items". The problem is that 0 is falsy.

If zero is a valid value in your context, check explicitly:

```
let quantity = 0
let message = quantity !== undefined ? quantity : "No items"

console.log(message)
```

Output: 0

Or use the nullish coalescing operator (??), which only treats null and undefined as falsy:

```
let quantity = 0
let message = quantity ?? "No items"

console.log(message)
```

Output: 0

The ?? operator is like || but only falls back for null and undefined, not for 0 or "".

PRACTICAL EXAMPLES

Form validation:

```
let email = "user@example.com"
let password = "secret123"

if (email && password) {
 console.log("Form is complete")
} else {
 console.log("Please fill in all fields")
}
```

Config with defaults:

```
let userSettings = {
 theme: "",
 fontSize: 0
}

let theme = userSettings.theme || "light"
let fontSize = userSettings.fontSize || 14

console.log(`Theme: ${theme}, Size: ${fontSize}`)
```

Output: Theme: light, Size: 14

(Note: fontSize 0 is probably a bug here. Use ?? if 0 should be valid.)

Safe property access:

```
let user = null

if (user && user.name) {
 console.log(user.name)
} else {
 console.log("No user")
}
```

Output: No user

Error messages:

```
let errorMessage = ""

if (!errorMessage) {
 console.log("No errors!")
}
```

Output: No errors!

The empty string is falsy, so !errorMessage is true.

PRACTICE

Check which of these are truthy: "false", 0, "0", null, [], "", {}, undefined, -1

Write code that uses a default greeting of "Hello" if no custom greeting is provided.

Write code that checks if both a username and email are provided (neither empty).

Write code that shows "Loading..." if a data variable is undefined or null.

WHAT YOU LEARNED

You learned that any value can be used as a condition.
You learned the six falsy values: false, 0, "", null, undefined, NaN.
You learned that everything else is truthy.
You learned to use || for default values.
You learned the difference between || and ?? for null-only checks.

You learned to watch out for valid zeros being treated as falsy.

The next chapter brings everything together with practice programs.

BOOK 2: MAKING DECISIONS
Chapter 8: Practice Decision Programs

LOGIN SYSTEM

Build a simple login checker with multiple validation steps:

```
let username = "admin"
let password = "secret123"
let storedUsername = "admin"
let storedPassword = "secret123"
let isAccountLocked = false
let loginAttempts = 2

console.log("=== LOGIN SYSTEM ===")

if (isAccountLocked) {
 console.log("Account is locked. Contact support.")
} else if (loginAttempts >= 3) {
 console.log("Too many failed attempts. Account locked.")
} else if (!username || !password) {
 console.log("Please enter both username and password.")
} else if (username !== storedUsername) {
 console.log("Username not found.")
} else if (password !== storedPassword) {
 console.log("Incorrect password.")
 let remainingAttempts = 3 - loginAttempts - 1
 console.log(`${remainingAttempts} attempts remaining.`)
} else {
 console.log(`Welcome back, ${username}!`)
 console.log("Login successful.")
}
```

Try changing the variables to see different messages. This demonstrates how real login systems work: check conditions in order, from most important to least.

TICKET PRICING SYSTEM

Calculate ticket prices based on age and membership:

```
let age = 25
let isMember = true
let isStudent = false
let dayOfWeek = "Tuesday"

let basePrice = 15
let finalPrice = basePrice
let discounts = []

console.log("=== MOVIE TICKET ===")
console.log(`Base price: $${basePrice}`)

if (age < 3) {
 finalPrice = 0
 discounts.push("Free (under 3)")
} else if (age < 12) {
 finalPrice = basePrice * 0.5
 discounts.push("Child discount: 50% off")
} else if (age >= 65) {
 finalPrice = basePrice * 0.7
 discounts.push("Senior discount: 30% off")
}

if (isMember && finalPrice > 0) {
 finalPrice = finalPrice * 0.9
 discounts.push("Member discount: 10% off")
}
```

```
if (isStudent && !isMember && finalPrice > 0) {
 finalPrice = finalPrice * 0.85
 discounts.push("Student discount: 15% off")
}

if (dayOfWeek === "Tuesday" || dayOfWeek ===
"Wednesday") {
 if (finalPrice > 0) {
 finalPrice = finalPrice - 2
 discounts.push("Weekday special: $2 off")
 }
}

console.log("")
if (discounts.length > 0) {
 console.log("Applied discounts:")
 for (let i = 0; i < discounts.length; i++) {
 console.log(`  - ${discounts[i]}`)
 }
}

finalPrice = Math.round(finalPrice * 100) / 100
console.log(`Final price: $${finalPrice}`)
```

GRADE CALCULATOR

Calculate a letter grade with plus and minus:

```
let score = 87

let letter
let modifier = ""

if (score >= 90) {
 letter = "A"
} else if (score >= 80) {
 letter = "B"
```

```
} else if (score >= 70) {
 letter = "C"
} else if (score >= 60) {
 letter = "D"
} else {
 letter = "F"
}

if (letter !== "F") {
 let lastDigit = score % 10

 if (score >= 90 && score <= 100) {
 lastDigit = score - 90
 } else if (score >= 80) {
 lastDigit = score - 80
 } else if (score >= 70) {
 lastDigit = score - 70
 } else if (score >= 60) {
 lastDigit = score - 60
 }

 if (lastDigit >= 7) {
 modifier = "+"
 } else if (lastDigit <= 2) {
 modifier = "-"
 }

 if (letter === "A" && modifier === "+") {
 modifier = ""
 }
}

console.log(`Score: ${score}`)
console.log(`Grade: ${letter}${modifier}`)

let feedback
if (letter === "A") {
 feedback = "Excellent work!"
} else if (letter === "B") {
 feedback = "Good job!"
```

```
} else if (letter === "C") {
 feedback = "Satisfactory. Keep studying."
} else if (letter === "D") {
 feedback = "Needs improvement."
} else {
 feedback = "Please see the instructor."
}

console.log(feedback)
```

SHIPPING CALCULATOR

Calculate shipping based on weight, destination, and speed:

```
let weightPounds = 3.5
let destination = "domestic"
let shippingSpeed = "express"
let isFragile = true

let shippingCost = 0
let estimatedDays = 0

console.log("=== SHIPPING CALCULATOR ===")
console.log(`Weight: ${weightPounds} lbs`)
console.log(`Destination: ${destination}`)
console.log(`Speed: ${shippingSpeed}`)
console.log(`Fragile: ${isFragile ? "Yes" : "No"}`)
console.log("")

if (destination === "domestic") {
 if (weightPounds <= 1) {
 shippingCost = 5
 } else if (weightPounds <= 5) {
 shippingCost = 8
 } else if (weightPounds <= 20) {
 shippingCost = 15
 } else {
```

```
 shippingCost = 25
 }

 if (shippingSpeed === "standard") {
 estimatedDays = 7
 } else if (shippingSpeed === "express") {
 shippingCost = shippingCost * 1.5
 estimatedDays = 3
 } else if (shippingSpeed === "overnight") {
 shippingCost = shippingCost * 3
 estimatedDays = 1
 }
} else if (destination === "international") {
 if (weightPounds <= 1) {
 shippingCost = 15
 } else if (weightPounds <= 5) {
 shippingCost = 30
 } else if (weightPounds <= 20) {
 shippingCost = 60
 } else {
 shippingCost = 100
 }

 if (shippingSpeed === "standard") {
 estimatedDays = 21
 } else if (shippingSpeed === "express") {
 shippingCost = shippingCost * 1.5
 estimatedDays = 10
 } else if (shippingSpeed === "overnight") {
 shippingCost = shippingCost * 2.5
 estimatedDays = 3
 }
}

if (isFragile) {
 shippingCost = shippingCost + 5
 console.log("Fragile handling fee: $5")
}

shippingCost = Math.round(shippingCost * 100) / 100
```

```
console.log(`Shipping cost: $${shippingCost}`)
console.log(`Estimated delivery: ${estimatedDays} days`)
```

ROCK PAPER SCISSORS

A simple game with all logic paths:

```
let playerChoice = "rock"
let computerChoice = "scissors"

console.log("=== ROCK PAPER SCISSORS ===")
console.log(`You chose: ${playerChoice}`)
console.log(`Computer chose: ${computerChoice}`)
console.log("")

let result

if (playerChoice === computerChoice) {
 result = "tie"
} else if (playerChoice === "rock") {
 if (computerChoice === "scissors") {
 result = "win"
 } else {
 result = "lose"
 }
} else if (playerChoice === "paper") {
 if (computerChoice === "rock") {
 result = "win"
 } else {
 result = "lose"
 }
} else if (playerChoice === "scissors") {
 if (computerChoice === "paper") {
 result = "win"
 } else {
 result = "lose"
```

```
 }
} else {
 result = "invalid"
}

if (result === "win") {
 console.log("You win!")
} else if (result === "lose") {
 console.log("You lose!")
} else if (result === "tie") {
 console.log("It is a tie!")
} else {
 console.log("Invalid choice. Use rock, paper, or scissors.")
}
```

TIME ZONE GREETING

Create greetings for different time zones:

```
let hour = 14
let timezone = "EST"
let userName = "Alex"

let adjustedHour = hour

switch (timezone) {
 case "PST":
 adjustedHour = hour
 break
 case "MST":
 adjustedHour = hour + 1
 break
 case "CST":
 adjustedHour = hour + 2
 break
 case "EST":
 adjustedHour = hour + 3
```

```
 break
 default:
 console.log("Unknown timezone, using local time")
}

if (adjustedHour >= 24) {
 adjustedHour = adjustedHour - 24
}

let greeting
let emoji

if (adjustedHour >= 5 && adjustedHour < 12) {
 greeting = "Good morning"
 emoji = "sunrise"
} else if (adjustedHour >= 12 && adjustedHour < 17) {
 greeting = "Good afternoon"
 emoji = "sun"
} else if (adjustedHour >= 17 && adjustedHour < 21) {
 greeting = "Good evening"
 emoji = "sunset"
} else {
 greeting = "Good night"
 emoji = "moon"
}

console.log(`${greeting}, ${userName}!`)
console.log(`Current time in ${timezone}: ${adjustedHour}:00`)
console.log(`Status: ${emoji}`)
```

PASSWORD STRENGTH CHECKER

Analyze password strength with detailed feedback:

```
let password = "MyP@ss123"

let length = password.length
```

```
let hasUppercase = password !== password.toLowerCase()
let hasLowercase = password !== password.toUpperCase()
let hasNumber = false
let hasSpecial = false

let numbers = "0123456789"
let specials = "!@#$%^&*()_+-=[]{}|;:,.<>?"

for (let i = 0; i < password.length; i++) {
 if (numbers.includes(password[i])) {
 hasNumber = true
 }
 if (specials.includes(password[i])) {
 hasSpecial = true
 }
}

console.log("=== PASSWORD STRENGTH ===")
console.log(`Password: ${password}`)
console.log("")

let score = 0
let feedback = []

if (length >= 8) {
 score = score + 1
 feedback.push("Length OK (8+ characters)")
} else {
 feedback.push("Too short (need 8+ characters)")
}

if (length >= 12) {
 score = score + 1
 feedback.push("Bonus: Long password (12+ characters)")
}

if (hasUppercase) {
 score = score + 1
 feedback.push("Has uppercase letters")
} else {
```

```
 feedback.push("Missing uppercase letters")
}

if (hasLowercase) {
 score = score + 1
 feedback.push("Has lowercase letters")
} else {
 feedback.push("Missing lowercase letters")
}

if (hasNumber) {
 score = score + 1
 feedback.push("Has numbers")
} else {
 feedback.push("Missing numbers")
}

if (hasSpecial) {
 score = score + 1
 feedback.push("Has special characters")
} else {
 feedback.push("Missing special characters")
}

console.log("Analysis:")
for (let i = 0; i < feedback.length; i++) {
 console.log(`  - ${feedback[i]}`)
}

console.log("")

let strength
if (score >= 6) {
 strength = "Very Strong"
} else if (score >= 4) {
 strength = "Strong"
} else if (score >= 3) {
 strength = "Medium"
} else {
 strength = "Weak"
```

```
}

console.log(`Score: ${score}/6`)
console.log(`Strength: ${strength}`)
```

WHAT YOU BUILT

You built seven complete programs demonstrating:

Login validation with multiple conditions
Price calculations with discounts and rules
Grade assignment with modifiers
Shipping cost calculations with nested logic
Game logic with win/lose/tie outcomes
Time-based greetings with zone adjustments
Password analysis with scoring

Each program combined:

Variables and data types from Book 1
If, else if, else chains
Nested conditions
Switch statements
Ternary operators
Truthy and falsy checks
Logical operators

WHAT COMES NEXT

Your programs can now make decisions. But they still run once and stop. They cannot repeat operations.

Book 3 teaches loops. With loops, you can:

Process every item in a list
Repeat until a condition changes
Count from one to a million
Search through data
Run the same code with different values

Decisions let programs think. Loops let programs work. Together, they create powerful software.

You finished Book 2. You can make programs that choose between paths. Now we make them repeat.